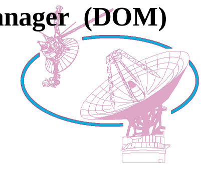


The Distributed Object Manager (DOM)

A Developer’s Perspective

Rick Borgen

22 September 2000

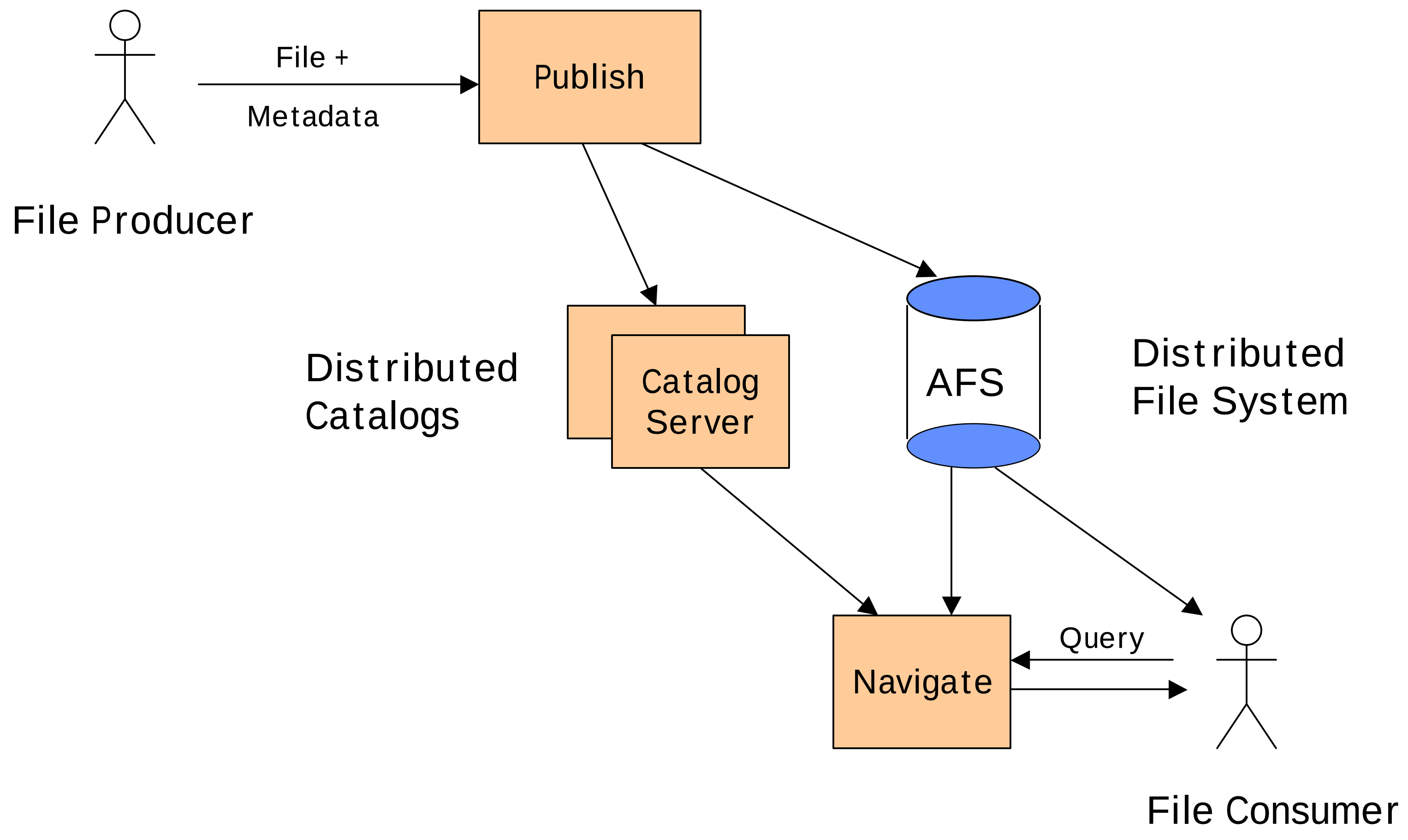
Distributed Object Manager (DOM)

- DOM is a general-purpose, extensible, distributed, object-oriented cataloging and file management system.

- DOM features lightweight, adaptable catalog servers with universal GUIs for query and file submittal. These enable rapid setup and deployment for novel applications/file types without the need for software changes.

- Implementation based on C++, Motif, Java, RMI, and distributed file systems (AFS, NFS, ...)

DOM File Management



DOM Customers

TMOD Deep Space Customers

- Past Projects
  - Mars Pathfinder
- Present Projects
  - Deep Space 1
  - Cassini
- Future Projects
  - SIRTF (prototype)
  - Mars '03 (prototype)
- Multi-Mission Applications
  - MSAS
  - Commanding
  - Data Quality Analysis & Verification (prototype)
  - Tracking Data and Delivery System (planning)

Science Customers

- GPS Genesis
  - Occultation products
  - Complex metadata
  - Very large populations
  - Internet distribution
- Atmosphere Infrared Sounder (AIRS)
  - Complex metadata
  - Very large populations
  - Principally internal use

DOM Developers

- Rick Borgen (SE)
- Dave Wagner (CDE)

- Edith Nir
- Dah Chang
- Ken Lam
- Bach Bui

DOM Origins

- **SFOC (now AMMOS) - a multi-mission ground system**
  – File management emerges as a key problem (a retreat from grand goal to put everything in a DBMS!)
  – Adaptable/configurable systems a driving SFOC architectural goal; but a hard problem for a catalog system
- **Planetary Data System (PDS) invents the PVL**
  – Parameter/Value Language used for labeling archive products
  – Simple, flexible, precise, human-readable, machine-readable
  – An object-centric description (not normalized!)
- **Object Technology matures**
  – C++ emerges as leading OO language
  – STL provides powerful micro-database library

Sample PVL Metadata Label

```
MISSION_NAME = CASSINI
SPACECRAFT_NAME = CASSINI
SPACECRAFT_ID = 82
DATA_SET_ID = TRACKING_DATA_FILE
FILE_NAME = xyz123.trk
PRODUCT_CREATION_TIME = 1998-204T20:04:02.000
PRODUCER_ID = DSN
DSN_STATION_NUMBER = {32,42}
DSN_SPACECRAFT_NUM = 182
START_TIME = 1998-204T12:05:07:32.003
STOP_TIME = 1998-204T20:15:11:30.322
```

A Search Engine

- **Dubbed “darkstar”: custom search engine, not a DBMS**
- **Goal 1: Support a hybrid of database ideas:**
  – Catalog objects are strongly-typed objects with PVL-like descriptions.
  – Collections are like relational tables with object members
  – Collections are arranged as hierarchies.
  – SQL-based search language (no joins)
  – Supports hierarchy of types and super-types
- **Goal 2: Provide a completely transparent schema**
  – Schema details enable adaptable, self-configuring applications
- **Based on C++/STL/yacc (~ 24K lines)**
- **Standard Template Library (STL) was a key technology**
  – provides a kind of micro-database technology in library form: vectors, sets, maps, sorting, ...

DOM Programs/Libraries

Motif GUI Programs

- catnav -Catalog Navigator
- catpub -Catalog Publisher
- catfm -Catalog File Manager
- catbrws - Catalog Browser
- catedit - Catalog Editor

Unix Command-lines

- darkcat - interactive DQL
- darkstar - catalog server
- cat_getfile
- cat_publish
- cat_replace
- cat_delete
- cat_catalog
- cat_superquery

C/C++ Libraries

- catSession
- libdql
- gui (catnav, catpub, catfm, catedit, catbrws)

Java GUI Programs

- catNavigator
- catPublisher
- catEditor
- catNotify (file notification subscription)
- easyQuery (custom query menus)

Java Command-lines

- catPublish
- catGetFile

Java Libraries

- dqlAPI
- catSession
- sssMetaData
- JDBCAdapter

Perl Libraries

- dqlAPI

DOM - File Systems

- **Old Rule**: DOM file system must be visible to ALL clients (e.g., AFS, DFS, NFS)
- **New Rule**: Web clients may access files remotely

- **DOM avoided the "vault" model (ALL access via catalog)**
- **File system is open for reads, but closed for writes**

- **TMOD systems use AFS**
  - wide distribution, effective security, backups
- **Science Customers use NFS**
  - cheap, high-capacity storage

The Art of the GUI

GUI Experiences

It is quite a trick to provide a rich, powerful set of functions AND keep things simple.

- **GUI design is an art.**
- **Less is sometimes more.**
- **There is no substitute for complete reliability.**
- **Actual usage will often surprise.**
- **Focus on a few key GUIs did pay off.**
- **For us, the schema design drives GUI look-and-feel**
- **Iterate, iterate, iterate.**

An Early Goof

The ability to rapidly generate collection trees was intoxicating - too often we made these more complex than warranted by actual usage

- **Collection tree heuristics**
  - No more than a few hundred collections per server
  - No more than a thousand items per collection
  - If collections remain empty…may need pruning
  - Collection populations should be balanced
- **Use the "garden walk" rule**
  - Don't pour the concrete until you see where they beat the paths

Java/Web Initiative

- Rebuilt (almost) all DOM clients in Java
- Using 3-tier architecture with RMI servers
- Aimed at deploying clients connected via the internet, assuming essentially no JPL infrastructure

- Web clients in limited deployment
- Authentication/security needs solving (in prototype)
- Look for broader deployment in 2001

Programming Languages

- **We like C++**
  - Very effective, though quite complex
  - Main problems due to a moving standard, variable compiler behavior, compiler/linker problems
- **But we love Java**
  - Very effective, simpler and safer
  - Fast enough, at least for our client-side work
  - Write once, run anywhere comes close
  - Main problems due to evolution of language, lags in support on some platforms
- **We prefer Java applications over applets.**
  - Complex GUIs do not coordinate so well with Web browsers…e.g., multiple sets of scroll bars
  - Some clients are command-line tools

JPL has built LOTS of Catalogs

- **File products are backbone of many JPL data systems…leads to need for catalogs**
  - files are a practical solution to high-volume, complex, technical data…full DBMS treatment is problematic
- **Many DBMS solutions tried for catalogs**
  - (e.g., Oracle, Sybase, Ingres, ObjectStore, Versant, LDAP, WAIS,…), but RDBMSs have been most popular
- **Schema sensitivity the biggest problem**
  - using standard DBMS the normal way means schema changes drive application changes…a huge barrier to effective re-use